Writing a C++ class for infinite-precision integers.

There are two files in this project (**ped1.cpp** and **inf_precision_Ped.hpp**).

A. ped1.cpp

This file contains a few examples of the of integer numbers showing how the results are reliable. To start with the first integer is "-1" and the second one is "1" (more results are provided in the appendix). Here are the results:

Table 1: num#1 =-1 and num#2 = 1

Num#1: -1	Num#1 **==** Num#2?
Num#2: 1	false
Num#1 **+** Num#2= 0	Num#1 **>=** Num#2?
Num#1 **-** Num#2= -2	false
Num#1 * Num#2= -1	Num#1 <= Num#2?
++Num#1 = 0	true
Num#1**++** = 1	Num#1 != Num#2?
--Num#1 = 0	true
Num#1**--** = -1	Num#1 *= Num#2= -1
Num#1 **>** Num#2?	Num#1 **-=** Num#2= -2
false	Num#1 **+=** Num#2= -1
Num#1 **<** Num#2?	**-**Num#2= -1
true	

Every number is saved in a vector with uint64_t. Although, the first digit of each vector represents + or – with 1 or 0, respectively. There is also an operator << when the input is **a inf_precision_Ped** class. Since the output of most operations are class, this operator for printing is developed.

B. inf_precision_Ped.hpp

The first constructor in the class receives the string and after checking all characters, it saves the string into a vector using vector.push_back(), if they are valid. The valid characters as input are "+", "- ", and numbers. The code also removes the initial unnecessary zeros. There are some functions, including **addingTwoPos()**, **subtracting()**, **multiplying()**, and **tasavi()** for comparison of the numbers, are provided to make the program less wordy! The function **get_vec()** is used to receive the vector of the class before doing any operations on vectors. The class **inf_precision_Ped(const vector<uint64_t> &_pedram)** is used to return the output of each operation in terms of class. The constructor **inf_precision_Ped()** with no input is defined to throw error when the input is empty, for example "". The function **changeV()** is developed only to update the values of the class after operations +=, -=, and *= are used.

In addingTwoPos() and subtracting() and multiplying() a carrier is considered to add the numbers. Here are the operations programed in the project:

- **inf_precision_Ped operator+(const inf_precision_Ped &_v, const inf_precision_Ped &_w);**

The output of this operator is a class. The function addingTwoPos() and subtracting() both are used to compute the results based on the comparison of the numbers using tasavi(). If x>0 and y >0, there are four possible scenarios for summation: x+y, x-y, -(x-y) and –(x+y).

- **inf_precision_Ped operator+=(inf_precision_Ped &_v, const inf_precision_Ped &_w);**

The previous operator is used to compute the results here. **The result is sent back to the function changeV() in the public class to modify the value.** In table 1, after using **-=,** the value of Num#1 is changed from -1 to -2. In the next line, the summation of Num#1+=Num#2 is actually -2+(1) = -1 which is the final value of Num#2.

- **inf_precision_Ped operator-(const inf_precision_Ped &_v, const inf_precision_Ped &_w);**

The same as operator+.

- **inf_precision_Ped operator-=(inf_precision_Ped &_v, const inf_precision_Ped &_w);**

The same as operator +=.

- **inf_precision_Ped operator*(const inf_precision_Ped &_v, const inf_precision_Ped &_w);**

In this operator the functions addingTwoPos() and multiplying() are used. First, the first digits of both numbers are changed to 1 (changing them to positive values). The sign of the results are decided based on the possible scenarios, including (+)*(+), (-)*(-), and (+)*(-),

- **inf_precision_Ped operator*=(inf_precision_Ped &_v, const inf_precision_Ped &_w);**

The same as operator +=.

- **inf_precision_Ped operator-(const inf_precision_Ped &_v);**

The first digit of the vector will be changed in this operator.

- **bool operator==(const inf_precision_Ped &_v, const inf_precision_Ped &_w);**

Using **tasavi()** function, the output will be -1 when the first number is smaller than the second number, +1 is the opposite, and 0 when both numbers are equal. Inside the operator the Boolean is given based on the output of this function.

- **bool operator<=(const inf_precision_Ped &_v, const inf_precision_Ped &_w);**

The same as == operator.

- **bool operator>=(const inf_precision_Ped &_v, const inf_precision_Ped &_w);**

The same as == operator.

- **bool operator!=(const inf_precision_Ped &_v, const inf_precision_Ped &_w);**

The same as == operator.

- **bool operator<(const inf_precision_Ped &_v, const inf_precision_Ped &_w);**

The same as == operator.

- **bool operator>(const inf_precision_Ped &_v, const inf_precision_Ped &_w);**

The same as == operator.

NOTE: The reference of the following operators: https://en.cppreference.com/w/cpp/language/operators

- **inf_precision_Ped &operator++()**

Based on the positivity or negativity of the number, addingTwoPos() and subtracting() are used to do prefix increment.

- **inf_precision_Ped operator++(int)**

The previous function is used here to operate postfix increment.

- **inf_precision_Ped &operator--()**

Based on the positivity or negativity of the number, addingTwoPos() and subtracting() are used to do prefix decrement.

- **inf_precision_Ped operator--(int)**

The previous function is used here to operate postfix decrement.

Appendix

Example 1:

Table 2: num#1 = +000065138464834800098656454517931564602317 and num#2 = -6236487630129

```
Num#1: 65138464834800098656454517931564602317
Num#2: -6236487630129
Num#1 + Num#2= 65138464834800098656454511695076972188
Num#1 - Num#2= 65138464834800098656454524168052232446
Num#1 * Num#2= -406235230187823670757447433464498461709011672408893
++Num#1 = 65138464834800098656454517931564602318
Num#1++ = 65138464834800098656454517931564602319
--Num#1 = 65138464834800098656454517931564602318
Num#1-- = 65138464834800098656454517931564602317
Num#1 > Num#2?
true
Num#1 < Num#2?
false
Num#1 == Num#2?
false
Num#1 >= Num#2?
true
Num#1 <= Num#2?
false
Num#1 != Num#2?
true
Num#1 *= Num#2= -406235230187823670757447433464498461709011672408893
Num#1 -= Num#2= -406235230187823670757447433464498461702775184778764
Num#1 += Num#2= -406235230187823670757447433464498461709011672408893
-Num#2= 6236487630129
```

Example 2:

Table 3: num#1 = +6236487630129 and num#2 = 65323fdfsd847

```
Num#1: 6236487630129
Error: The entered value is not an integer!
```

Example 3:

Table 4: num#1 ="" and num#2 = 1

```
Error: The entered value is not an integer!
```

To get more results, please change the str_Ped and str_Ped2 in the **ped1.cpp** file.